EXHIBIT 1
Part 2 of 4

This petition explains why the claims of the '526 patent are unpatentable over the prior-art Martinez-Guerra patent, applied for in February 1999, which is described in more detail in Section VIII below.

V. LEVEL OF ORDINARY SKILL IN THE ART

A person of ordinary skill in the art ("POSA") is presumed to be aware of all pertinent art, follows the conventional wisdom in the art, and is a person of ordinary creativity. With respect to the '526 patent, a POSA would typically have at least a Bachelor's Degree in computer science and 3-5 years of experience in systems development. (Clark Decl. ¶¶ 15-16.)

VI. CLAIM CONSTRUCTION

The Board gives claims their broadest reasonable interpretation ("BRI") consistent with the specification. 37 C.F.R. § 42.100(b); *In re Cuozzo Speed Technologies, LLC*, 793 F.3d 1268, 1279 (Fed. Cir. 2015).[1] Such a construction must reasonably reflect the plain language and disclosure of the patent "as [they]

[1] Other forums, such as district courts, apply a different standard. *See Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (*en banc*). Any interpretation of the challenged claims in this petition, either explicit or implicit, does not reflect Arista's interpretation under a standard other than the BRI. In particular, Arista reserves the right to argue for different constructions under the *Phillips* standard in the related district-court litigation.

would be interpreted by one of ordinary skill in the art." *In re Suitco Surface Inc.*, 603 F.3d 1255, 1259 (Fed. Cir. 2010).

As relevant here, the Board should construe the claim terms as follows:

A. "management program"

Proposed construction: "separate tools or external agents having their own respective command formats that provide management functions."

Cisco Systems, Inc. ("Cisco"), the owner of the '526 patent, has proposed the above construction for "management program" in the related litigation. (Cisco Preliminary Claim Construction Disclosure, *Cisco Systems, Inc. v. Arista Networks, Inc.*, No. 5:14-cv-05344-BLF (N.D. Cal. Aug. 24, 2015) (Ex. 1010) at 3.) For the purpose of this proceeding, Arista agrees that Cisco's construction would be the broadest reasonable interpretation of this term. The '526 patent states that management programs "may be executed within the processor based system or externally as external agents," and describes them as having their own "respective command formats and syntax." (Ex. 1001 at 3:1-5, 1:41-44; *see also* Clark Decl. ¶ 29.)

B. "command parse tree"

Proposed construction: "a hierarchical data representation having elements each specifying at least one corresponding generic command component and a corresponding at least one action value."

Cisco has proposed the above construction for "command parse tree" in the related litigation. (Ex. 1010 at 10.) For the purpose of this proceeding, Arista agrees that Cisco's construction would be the broadest reasonable interpretation of this term. Although a "parse tree" commonly refers to the result of parsing an input stream into its constituent parts, a skilled artisan would understand that the broadest reasonable interpretation of the phrase "command parse tree" in the '526 patent is not the result of a parsing operation; it is instead a tree representation of the entire language, used to perform a parsing operation, as illustrated in Figure 2 of the patent. (*See* Ex. 1001 at 3:39 (referring to "command parse tree 22"); *see also* Clark Decl. ¶ 30.)

C. "recursively traversing"

Proposed construction: "traversing using a process that repeats itself" The '526 patent uses the term "recursively traversing" only once outside the claims, stating that "parser 14 recursively traverses the command parse tree 22 for each command word to identify the best match for the generic command." (Ex. 1001 at 3:55-57.) Figure 3 and the descriptions thereof explain the process of "traversing" the command parse tree: the parser repeats a series of steps in a looped manner for each successive command word. (*See, e.g.*, *id.* at 4:10-11, 4:19-23, Fig. 3.) Because Figure 3 shows a process by which a set of steps is repeated in a looped manner, one of skill in the art would understand that the broadest

reasonable interpretation of "recursively traversing" includes such a looping mechanism. (Clark Decl. ¶¶ 31-32.)

D. "means for validating" / "validating means"

This claim phrase is subject to 35 U.S.C. § 112(f) (formerly 35 U.S.C. § 112, ¶ 6).

Function: "validating a generic command received from a user."

Corresponding Structure: Parser 14 of Figures 1 and 2, as described in 3:36-61, and executing an algorithm as disclosed in Figure 3.

The '526 patent states that "[t]he parser 14 is configured for validating a received generic command by comparing each input command word to the command parse tree 22 to determine for the received generic command a tree element 24 identified as a best match." (Ex. 1001 at 3:47-51; *see* also Clark Decl. ¶¶ 33-34.)

VII. SUMMARY OF THE PRIOR ART FORMING THE BASIS OF THIS PETITION

U.S. Patent No. 6,523,172 to Martinez-Guerra *et al.*, entitled "Parser Translator System and Method" ("Martinez-Guerra") (Ex. 1002), was filed February 19, 1999, and issued February 18, 2003. Martinez-Guerra is thus prior art to the '526 patent under 35 U.S.C. § 102(e).

Martinez-Guerra, which was not before the Office during prosecution of the '526 patent, is directed to a "parser-translator technology" that allows users to

enter statements "in a high-level language," and translates those statements into "directives appropriate to a particular data processing application." (Ex. 1002, Abstract; *see* also Clark Decl. ¶ 36.) As Martinez-Guerra explains, "[u]sing the parser-translator technology, a user can focus on the semantics of the desired operations and need not be concerned with the proper syntax of a language for a particular system." (Ex. 1002, Abstract.)

Martinez-Guerra teaches that its parser-translator can be used to control multiple programs, each with its own grammar. (*Id.*) For example, it teaches that "a single parser-translator component provides multiple software applications with an interface to high-level user language statements wherein operation of the single parser-translator component is suitably defined for each software application using a corresponding grammar encoding." (*Id.* at 3:48-53; *see also, e.g., id.* at 13:34-42 (describing how "tool-specific translation rules" enable "a parser-translator component implemented in accordance with the present invention [to] be used to provide a natural language dialogue facility in a wide variety of software tool environments."); Clark Decl. ¶ 36-37.)

With respect to the parser itself, Martinez-Guerra teaches that "[a] variety of token recognizer . . ., parser . . ., and translator . . . designs are suitable," and describes exemplary embodiments. (Ex. 1002 at 10:31-41.) The embodiments employ a grammar that includes "phrase structure rules," which "describe the legal

ordering of tokens in a language"; "dictionary entries," which "describe all the tokens that a parser-translator . . . should recognize"; and "mapping rules," which "indicate how a sequence of tokens should be translated by the parser-translator[.]" (*Id.* at 14:8-21; *see* also Clark Decl. ¶ 38.) The "mapping rules" are also referred to as "translation rules." (Ex. 1002 at 15:23.)

As each word in a command is parsed, these rules define a set of valid next tokens in light of the tokens that have already been validated. (*See, e.g., id.* at 18:36-40 ("Token recognizer 31 computes a set of valid next states . . . [that] includes the set of next tokens consistent with a current parse state."); *see* also Clark Decl. ¶ 45.) At first, the set of valid tokens includes all "tokens that may begin a legal statement in the language defined by the grammar." (Ex. 1002 at 18:40-42.) Each successive token is then compared to the set of valid next tokens in light of the tokens that came before. (*See, e.g., id.* at 20:31-34 ("Token recognizer 31 performs input matching against each of the valid next states corresponding to a parse state representation consistent with the input stream read so far."); *see* also Clark Decl. ¶ 45.) Finally, the complete command is translated and the translated command is issued to the appropriate program. (*See, e.g.,* Ex. 1002 at 20:61-21:10 and Fig. 1.)

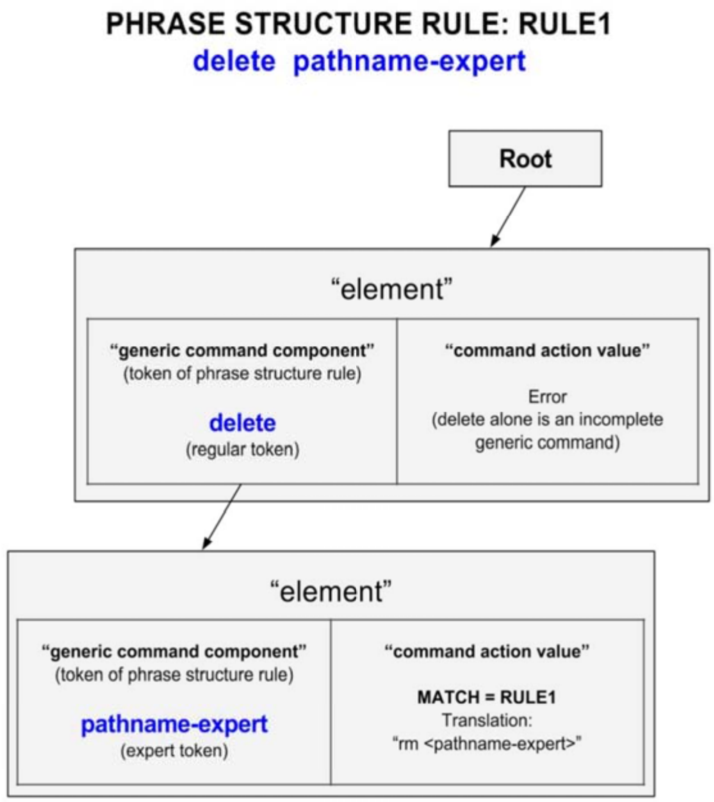
As an illustrative example, Martinez-Guerra explains how a parser-translator would translate the command "delete /usr/extract/testing" received from a user,

where "/usr/extract/testing" is the name of a file. (*See* Ex. 1002 at 15:50-16:5.) The parser-translator uses the hypothetical phrase structure rule "RULE1," which describes the legal ordering of tokens: "delete pathname-expert." (*Id.*; *see* also Clark Decl. ¶ 41.) "[D]elete is a regular token matching the string delete in the input stream[.]" (Ex. 1002 at 15:58-59.) "/usr/extract/testing" is what the patent calls an "expert token," which is "a string whose value cannot be specified when the grammar is written but which can be defined in terms of its characteristics." (*Id.* at 15:4-6, 15:67-16:5.) Thus, "pathname-expert" corresponds to the pathname of a file. (Clark Decl. ¶ 42.)

When the command "delete /usr/extract/testing" is received, each word is matched to a corresponding token found in the dictionary entries. (Ex. 1002 at 9:34-38, 14:60-15:15; Clark Decl. ¶ 42.) Martinez-Guerra explains that the dictionary entries identify all tokens that the parser-translator should recognize; thus, they act as a table for correlating an input word with a corresponding token. (*Id.* at 14:16-18, 14:60-62, 20:22-24; Clark Decl. ¶ 38, 41.) The tokens are then compared to the phrase structure rules. (*Id.* at 11:51-60; Clark Decl. ¶ 40-41.) Because "/usr/extract/testing" is recognized as a file pathname, the parser-translator will look for "pathname-expert" in place of "/usr/extract/testing." (*Id.* at 15:54-16:5; Clark Decl. ¶ 42.)

If the sequence of tokens matches a sequence in a phrase structure rule, then that rule is satisfied, and the parser-translator issues the corresponding translation. (Ex. 1002 at 9:40-43, 20:45-48; Clark Decl. ¶ 43.) In the example above, because the tokens "delete" and "pathname-expert" satisfy RULE1, the translator (if in a UNIX environment) translates the command as "rm <pathname-expert>," where "<pathname-expert>" will be replaced with the original user input. (Ex. 1002 at 15:59-16:5; Clark Decl. ¶ 43.) Thus, the parser-translator issues the prescribed command "rm /usr/extract/testing," directing the system to remove (*i.e.*, delete) the specified file. (Ex. 1002 at 16:4-5; *see also* Clark Decl. ¶ 43, 61.)

For any finite command language as described in the '526 patent, which are a subset of the languages addressed by Martinez-Guerra, Martinez-Guerra's phrase structure rules constitute a command parse tree because they are collectively a hierarchical data representation of the valid sequences of components of statements in a high-level user language, and because every component of each phrase structure rule has an action that will occur if that component is the last valid component parsed. For example, the phrase structure rule RULE1 would be represented as a set of component-action value pairs (what the '526 patent calls "elements") within such a command parse tree as illustrated in the figure below:



(Clark Decl. ¶¶ 38-39.)

VIII. IDENTIFICATION OF CHALLENGE (37 C.F.R. § 42.104(b))

Petitioner requests *inter partes* review and cancellation of claims 1-26 on the grounds that they are obvious over Martinez-Guerra. As set forth below, Petitioner believes that Martinez-Guerra discloses every limitation of claims 1-6, 8, 10-19, 21, and 23-26. However, because some disclosures may be seen as requiring corroboration from the expert, Petitioner challenges these claims under obviousness rather than anticipation. Claims 7, 9, 20, and 22 recite a single limitation that Martinez-Guerra does not expressly disclose, but that was

unquestionably obvious: that the management program *executes* the prescribed command it receives.

In support of the grounds for unpatentability, this petition is accompanied by the Declaration of Dr. Douglas Clark (Ex. 1014), who explains what Martinez-Guerra would have conveyed to a POSA, and why Martinez-Guerra discloses or renders obvious every challenged claim element.

A. Claim 1 and dependent claims 2-9 are invalid as obvious over Martinez-Guerra

1. [1A] "A method in a processor-based system configured for executing a plurality of management programs according to respective command formats, the method comprising"

Martinez-Guerra discloses a parser-translator that enables a high-level user language to interface with multiple software applications. (Ex. 1002 at 3:48-53; Clark Decl. ¶ 47.) It does so by translating statements that a user enters in a high-level language (which can include generic commands) to "logically and syntactically correct directives for performing the desired data transformations or operations" (prescribed commands) for the proper software tool. (Ex. 1002 at 3:29-37; Clark Decl. ¶ 47.) The method is carried out in software executed on a computer system. (Ex. 1002 at 9:10-13; Clark Decl. ¶ 47.)

The broad disclosure of software tools that can be used with the system includes "management programs." For example, Martinez-Guerra explains that "an illustrative enterprise tools environment includ[es] data discovery and cleansing

tools, data extraction conversion and migration tools, data movement and replication tools, query Multi-Dimensional Data (MDD) analysis and On-Line Analytical Processing (OLAP) tools, as well as applications and gateways that may advantageously incorporate a parser-translator component." (Ex. 1002 at 11:36-47; *see also id.* at 13:14-20 (additional software applications, including "query and on-line analytical processing tools, gateways to operational data systems"); Clark Decl. ¶ 48.)

The disclosed tools are separate from the parser-translator and from one another, and as illustrated by Martinez-Guerra's example of translating a "delete" command, use a command format that may differ from the high-level user language command provided by the user in the input stream. (*See* Ex. 1002 at 15:50-62 (translating the high-level input command "delete /usr/extract/testing" to the prescribed UNIX command "rm /usr/extract/testing"); *see also, e.g., id.* at 3:48-53 (describing a "corresponding grammar encoding" for "each software application"), 13:34-42 (describing "tool-specific translation rules," indicating that the tools have respective command formats); Clark Decl. ¶ 49.)

2. [1B] "receiving a generic command from the user;"

Martinez-Guerra discloses a user interface that receives from a human user an input stream consisting of statements in a "high-level user language[.]" (Ex. 1002, Abstract; *see also id.* (users can focus on "the semantics of the desired

operations" when inputting commands), Fig. 3 (item 36) (input stream), 9:15-23 (input "supplied by a human user"); Clark Decl. ¶ 50.) One of ordinary skill in the art would understand that such statements, as described in Martinez-Guerra, can be "generic commands" within the meaning of the '526 patent, because they are abstractions of desired operations that are later translated into directives for specific software tools. (Clark Decl. ¶ 51; *see also* Ex. 1002 at 10:7-11, 13:30-42.) For example, one of ordinary skill in the art would understand that the command "delete /usr/extract/testing" in Martinez-Guerra is analogous to the "Generic Command Examples" disclosed in the '526 patent. (*See* Ex. 1002 at 15:50-62; Ex. 1001 at 7:1-9:15 (identifying, *e.g*, "set watchtime <milliseconds>" as an exemplary generic command).)

3. [1C.1] "validating the generic command based on a command parse tree that specifies valid generic commands relative to a prescribed generic command format,"

Martinez-Guerra discloses the use of a command parse tree as claimed in the '526 patent. Martinez-Guerra explains that words in the input stream are analyzed by the token recognizer subcomponent of the parser-translator according to rules encoded in the grammar. (*See* Ex. 1002 at 10:42-58; Clark Decl. ¶ 52.) Martinez-Guerra defines "grammar" as "a formal declaration of the syntactic structure of a language typically represented as a set of rules that specify legal orderings of constituents and subconstituents (*e.g.*, phrases and symbols) in a language."

(Ex. 1002 at 7:52-56.) The "legal orderings" refer to valid sequences of tokens as defined by phrase structure rules in the grammar. (*Id.* at 8:24-26.) The grammar also provides translation functions corresponding to all such valid sequences of tokens. (*Id.* at 16:15-16, 17:51-52; Clark Decl. ¶ 52.)

Martinez-Guerra explains that the parser-translator creates "internal data structures," including "parse state data structures" and "internal representations of a grammar," which it uses to specify "next legal choices" allowed by the grammar. (*Id.* at 18:31-40 and 10:45-46; Clark Decl. ¶ 53.) "[T]he set of valid next states includes the set of next tokens consistent with the current parse state." (Ex. 1002 at 18:38-40.) As each token is ingested, the parser-translator uses the phrase structure rules encoded in the grammar to specify the possible valid next legal choices until the input stream has been fully parsed. (Clark Decl. ¶ 53; Ex. 1002 at 10:42-58, 20:45-49.) For any finite high-level command language, these phrase structure rules encoded in the grammar thus constitute a command parse tree, because they are collectively a hierarchical data representation of the valid components of statements in a high-level language, and as explained below, their elements specify at least one corresponding generic command component and a corresponding action value. (Clark Decl. ¶¶ 21-24, 53.)

When the input stream contains a generic command, the command is validated by the parser-translator relative to a prescribed generic command format

defined by the grammar. (*Id.* at 10:42-58; Clark Decl. ¶ 54.) Specifically, the token recognizer subcomponent receives the initial command word (*e.g.*, "delete") and first determines whether the word matches a token in the dictionary entries, which define all tokens found in the phrase structure rules of the grammar. (Clark Decl. ¶ 54; Ex. 1002 at 9:34-38, 14:16-18.)

Once it has confirmed a match, the token recognizer then determines whether the token corresponding to the command word is present among the next legal parse states maintained by the parser. (Clark Decl. ¶ 54; Ex. 1002 at 9:34-40, 20:28-34.) As discussed above, each parse state identifies valid next legal tokens, as defined by the phrase structure rules. (Clark Decl. ¶ 55.) In the example of column 15:50-62, the tokens "delete" and "pathname-expert" are matched, respectively, with the first and second legal parse words of the phrase structure rule, "RULE1." (Clark Decl. ¶ 55; Ex. 1002 at 15:50-62 ("RULE1 → delete pathname-expert").) Because the tokens "delete" and "pathname-expert," in that order, match the phrase structure rule RULE1, the generic command is validated. (Clark Decl. ¶ 55.)

4. [1C.2] "the command parse tree having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value,"

As discussed above, Martinez-Guerra explains that the grammar includes "a set of phrase structure rules and associated translation rules that define the

syntactic order of a source and target language . . . ." (Ex. 1002 at 7:57-59; *see also* 16:15-16.) And for any finite command language, the phrase structure rules encoded in the grammar constitute a command parse tree, because they are collectively a hierarchical representation of the valid components of statements in a high-level language, and as discussed below, their elements specify at least one corresponding generic command component and a corresponding action value. (Clark Decl. ¶¶ 21-24.) Phrase structure rules define a legal ordering of tokens in a particular language (*id.* at 14:11-12), and can be represented as textual rule specifications (*e.g.*, "RULE1 → delete pathname-expert"). (Clark Decl. ¶ 56.) In the textual rule specification for RULE1, each token to the right of the arrow is a component needed to form a complete statement; each of them is what the '526 patent calls a "generic command component." (*Id.*)

As claimed in the '526 patent, Martinez-Guerra discloses that each component of the phrase structure rule has a corresponding appropriate action that will occur if that component is a "best match" for the generic command. If an input word is invalid or a command is incomplete—for example, if only the word "delete" is received without any pathname to define the target of the operation—that action may result in returning an error. (*See* Ex. 1002 at 19:3-4.) On the other hand, a valid input stream that matches and completes a phrase structure rule (*e.g.*, "delete /usr/extract/testing" (*id.* at 16:5)) will correspond to an action of the

translation associated with the phrase structure rule. (Clark Decl. ¶ 57; Ex. 1002 at 8:60-65; *see also id.* at 14:19-21, 17:47-58.)

The instruction to return an error that results from inputting an invalid word or incomplete sequence (*e.g.*, inputting "delete" without a target pathname), or the translation function that results from inputting a complete and valid sequence of tokens (*e.g.*, "delete /usr/extract/testing"), is what the '526 patent refers to as a "command action value." Such a command action value exists for each phrase structure rule in Martinez-Guerra, and for each generic command component of a given phrase structure rule, because as input words are processed, one may be invalid, resulting in an error, or all may be valid, leading ultimately to the satisfaction of a phrase structure rule, thus resulting in a translation according to the corresponding translation rule. (Clark Decl. ¶ 57.) Each legal choice in the parse state is thus an "element," as described in the '526 patent, having both a generic command component and a corresponding command action value. (*Id.* at 58.)

5. [1C.3] "the validating step including identifying one of the elements as a best match relative to the generic command; and"

In the generic command example discussed above, Martinez-Guerra discloses that the generic command "delete /usr/extract/testing" is validated by the parser-translator because it matches the legal ordering of tokens described by the

phrase structure rule RULE1. (Clark Decl. ¶ 59; Ex. 1002 at 15:50-62.) In doing so, the parser-translator identifies the final element ("pathname-expert") of the rule specification RULE1 as the "best match" relative to the generic command because it is the final input word necessary to satisfy the phrase structure rule, thus causing the parser-translator to issue a prescribed command. (Clark Decl. ¶ 59.) As Martinez-Guerra states, "[o]nce the current parse state includes a complete phrase as defined by the phrase structure rules encoded in grammar 37, translator 33 provides a translation." (Ex. 1002 at 11:57-60.)

6. [1D] "issuing a prescribed command of a selected one of the management programs according to the corresponding command format, based on the identified one element."

Martinez-Guerra discloses that when a phrase structure rule is satisfied by the sequence of words received in the input stream, the translation function associated with that phrase structure rule causes the parser-translator to perform the translation. (Clark Decl. ¶ 60-61; Ex. 1002 at 11:55-62, 14:10-21, 17:47-55, 20:61-64.) The translator outputs the translation in the format used by the appropriate software tool. (*Id.* at 13:30-33 ("[T]he translation functionality of parser-translator component 30 provides the capability of translating the now properly formed user language statements *to the form and syntax desired by a specific software tool*.") (emphasis added); *see also id.* at 15:59-62, 21:33-36.)

For example, in the case of the generic command "delete /usr/extract/testing," after matching the token corresponding to the final generic command word with the next legal choice in the parse state ("pathname-expert"), the parser-translator carries out the translation using the translation function associated with phrase structure rule "RULE1," resulting in the prescribed UNIX command "rm /usr/extract/testing." (*Id.* at 15:59-62.) One of ordinary skill in the art would recognize that "rm" is the UNIX shell command used to delete files in a computer system. (Clark Decl. ¶ 61.) Martinez-Guerra thus discloses issuing a prescribed command for removing a file in a selected one of the management programs (a UNIX shell) according to the corresponding command format for that program ("rm /usr/extract/testing"). (Ex. 1002 at 15:50-62, 16:5.)

7. [2A] "The method of claim 1, wherein the generic command includes at least one input command word, the validating step including:"

Martinez-Guerra discloses a generic command (*e.g.*, "delete /usr/extract/testing,") which includes at least one input command word. (*Id.* at 15:50-62 (explaining that "delete" in RULE1 is a token "matching the string delete in the input stream such as input stream 36").)